

CSE 127

Week 2 Discussion

Agenda For Today:

- Setup VM
- VSCode setup
- GDB
- x86

PA1: GDB

- Thursday, October 19th at 11:59 pm
- Group submissions - Team size up at most 2
- Goal is to prepare you for the next assignment: Virtual machine and GDB

Virtual Machine

- Weird stack trace on startup and system doesn't start (AMD chips)
 - In advanced boot options, try booting using sysvinit or switch to an older kernel
- VirtualBox throws an error on startup
 - This varies, but on windows it is most likely because you haven't enabled Hyper-V, which there are resources to do [here](#). BIOS
- To make SSH work within WSL or other OS you can enable [port-forwarding](#).
- Use QEMU for M1 macs.
- SSH is not required, but you need a way to transfer your solution out of the VM.

GDB



- GNU Debugger
- Allows you to "see" inside your program
 - See registers, memory access, instructions
 - Breakpoints allow you to pause execution at any point

GDB Demo

GDB

- layout next
 - shows the code. To quit ctrl x + a
- b main
 - add breakpoint
- info frame
 - print info about the current stack frame
- info registers
 - registers in both hex and natural format
- x/[count][format][unit]
 - defaults: count = 1, format = x, unit = w
- x/10x \$ebp+4
 - show as hex
- x/10i \$eip
 - show as instructions
- x/5c name
 - show as char
- More uses of x [here](#)

GDB

- disass main → disassemble a function
- tui enable/disable → enable/disable text user interface (gdb -tui)
- layout src/asm → show source code/assembly
- tui reg general → show registers
- print i → prints variable value
- print i=10 → sets variable value
- set \$ebp = 123 → set value for a register
- set {int}0xffff12345 = 123 → set value for a memory region

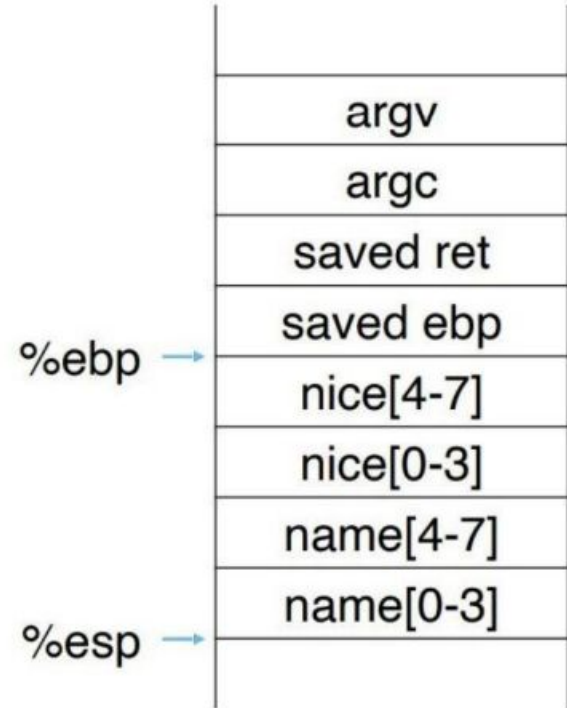
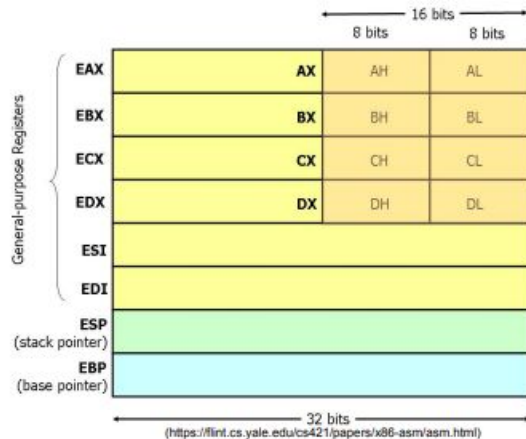
More resources [here](#)

Submission

- Submit hw1.txt to "PA1: gdb" on Gradescope.
- Follow the directions in the template, do not delete the square brackets
- If you remove the square brackets, the autograder won't recognize your answers.

x86 Registers

- `%esp`, or the Stack Pointer
 - Designates the top of the stack
 - Grows from high to low memory addresses
- `%ebp`, or the Frame Pointer/Base Pointer
 - Points to middle of stack frame(to the saved base pointer)
 - Doesn't move as function calls are made



x86 Registers

- %eip, or the Instruction Pointer
 - Holds the address of the next instruction to be executed

%eip →

```
pushl    %ebp
movl     %esp, %ebp
subl     $40, %esp
movl     16(%ebp), %eax
movl     %eax, -28(%ebp)
movl     %gs:20, %eax
movl     %eax, -12(%ebp)
xorl     %eax, %eax
```

x86 Registers

- `inc %eax` $\rightarrow$ `eax`
- `inc (%eax)` $\rightarrow$ `*eax`
- `inc 4(%eax)` $\rightarrow$ `*(eax + 4)`
- `inc 4(%eax, %ebx, 2)` $\rightarrow$ `*(eax + 4 + %ebx * 2)`

<https://patshaughnessy.net/2017/1/20/pointers-in-c-and-x86-assembly-language>

x86 Instructions

- movl
- cmpb
- je, jne, jmp
- add, sub, inc, dec
- int 0x80 - ?

x86 Instructions

- Byte (B)
 - 8-bits
- Word (W)
 - 16-bits = 2 bytes
- Double word (L)
 - 32-bits = 4 bytes
- Quad word (Q)
 - 64-bits = 8 bytes